Array: A set of the same data type where multiple data can be used with one name. A collection of several variables.

- The size of arrangement is set in the beginning and cannot be changed
- ORDER: Declare array \rightarrow allocate memory for an array \rightarrow use an array

1. OTHER FORMS:

1.1 Array type string:

```
String []varName = new String []
```

1.2 Array type int with values prescribed:

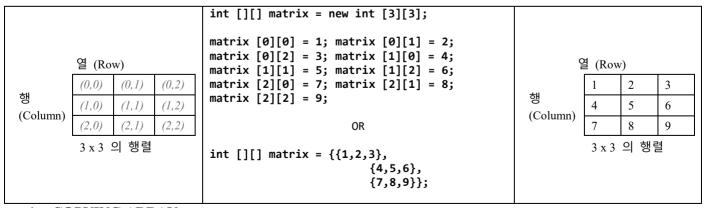
1.3 Array type objName with multiple values prescribed:

2. INSERTING VALUES IN THE ARRAY:

2.1 Integer

2.2 String

3. MULTIDIMENTIONAL ARRAY



1. COPYING ARRAY

Method 1:	Method 2:	Method 3:
Copying array 'score' to a new	Copying array 'score' to a new	Copying array 'score' to a new
variable named 's' using the <u>for</u>	variable named 's' using the <u>array</u>	variable named 's' using the
<u>command</u>	copy class.	

	In System.arraycopy: - score: original variable - 0: the initializing index value - s: copied variable - 0: from which index (in s) will it insert the copied values - score.length: the length of the array that is to be copied.	Copyof() method from the class Arrays.
<pre>int[] score = {100,10,20,30,40}; int[] s = new int[score.length];</pre>	<pre>int[] score = {100,10,20,30,40}; int[] s = new int[score.length];</pre>	<pre>int[] score = {100,10,20,30,40}; int[] s = null;</pre>
<pre>for(int idx=0; idx<score.length; idx++)="" idx,="" idx,<="" pre="" s[%d]='%d\n",' s[idx]="score[idx];" score[idx],="" system.out.printf("score[%d]="%d\t" {="" }=""></score.length;></pre>	<pre>System.arraycopy(score, 0, s, 0, score.length); for(int idx=0; idx<score.length; idx++)="" system.out.printf("score[%d]="%d\t</pre" {=""></score.length;></pre>	<pre>s = Arrays.copyOf(score, score.length); for(int idx=0; idx<s.length; idx++)="" idx_score[idx]_idx_s[idx]);<="" pre="" s[%d]='%d\n",' system.out.printf("score[%d]="%d\t" {=""></s.length;></pre>
s[idx]); }	s[d]=%d\n", idx, score[idx], idx, s[idx]);	<pre>idx, score[idx],idx,s[idx]); }</pre>
Output:	score [0] = 100 s[0] = 100 score [1] = 10 s[1] = 10 score [2] = 20 s[2] = 20 score [3] = 30 s[3] = 30 score [4] = 40 s[4] = 40	

2. IMPROVED FOR COMMAND

- Allows the program to read the command but not change in (increment or decrement it). Starting from the value in the first array, each array value gets printed in a variable defined by the programmer.
- Example 1: 65 is inputted in the variable "number" and this is printed, then 235 is inputted in the variable "number" and this is printed...etc...)
- For the improved for command, you cannot change a value in the array, whereas with the normal for command, you can. (See example 2)

Example 1:						Example 2:
	+	[] array : for (int : em.out.pr	number :	array)	{	<pre>int[] arr = new int[3]; for(int i=0 ; i<arr.length :="" ;="" arr)="" arr[i]="5;}" for(int="" i++)="" temp="9;}</pre" {=""></arr.length></pre>
						<pre>for(int temp : arr) { System.out.println(temp);</pre>
Output:		235	32	58	92	Output: 5 5 5 5

3. REFERENCE TYPE

Primitive Type: Store literal value of an integer, a real number, or a literal data type

String name1 = "Amy";

Reference Type: Does not store literal value but stores the address of where the literal value is stored.

String name1 = new String("Amy");

- When a reference type variable is created, the variable will have a null value
- When you make a variable with the same address (int[] variable1 = variable2;) and change something in either variable, the other variable also changes value in the same way, as they share the same address. (if an array is copied, they do not share the same address).

Comparing memory address:	Comparing the actual String value:
<pre>String name1 = new String("Amy"); String name2 = new String("Amy"); System.out.println(name1 == name2);</pre>	<pre>String name1 = new String("Amy"); String name2 = new String("Amy"); System.out.println(name1.equals(name2));</pre>
false	true